Interpretable privacy with optimizable utility

Jan Ramon, Moitree Basu

Interpretable privacy with optimizable utility

Jan Ramon and Moitree Basu

INRIA, Lille, France

Abstract. In this position paper, we discuss the problem of specifying privacy requirements for machine learning based systems, in an interpretable yet operational way. Explaining privacy-improving technology is a challenging problem, especially when the goal is to construct a system which at the same time is interpretable and has a high performance. In order to address this challenge, we propose to specify privacy requirements as constraints, leaving several options for the concrete implementation of the system open, followed by a constraint optimization approach to achieve an efficient implementation also, next to the interpretable privacy guarantees.

Keywords: Privacy · Explainability · Constraint optimization

1 Introduction

Over recent years, one has seen an increasing interest in privacy, as the awareness of privacy risks of data processing systems increased. Legislation was introduced to protect data, and sufficient data and insights became available to create technology capable to realize several tasks while preserving the privacy of participants. One of the most popular notions of privacy, which we will also adopt in this paper, is differential privacy [3] and its extensions, e.g., [4, 10].

An important aspect of this evolution is the question of informing users of what privacy guarantees a system offers. Some legislation, such as Europe's GDPR [1], requires transparency, i.e., users have the right to know how their data are used and how their sensitive data are protected. Explaining privacy protection strategies is also important to increase trust among the users of a system. Finally, being able to explain what privacy guarantees a system offers is also helpful in the sometimes challenging communication between computer scientists who develop solutions and legal experts who are interested in understanding the guarantees without the burden of having to investigate many technical details.

While a large number of papers in the machine learning community studies a single machine learning problem and strategies to perform that machine learning task in a privacy-preserving way, real-world systems are often complex, consisting of several machine learning, preprocessing, prediction or inference steps, user interactions, and data transfers. The privacy requirements of interest to a user are requirements on the system as a whole, combining the behavior of its many

components including their privacy guarantees. While some researches have focused on analyzing privacy guarantees of complete systems, the literature on that topic is still rather limited.

Such large systems combine heterogeneous components, each having their own characteristics for what concerns their effects on the privacy of the data. There is an increasing need for systems allowing one to specify and explain the privacy guarantees for a complete system. However, next to interpretability, performance, e.g., in terms of precision of computation, communication, and storage cost, is also required. In this paper, we study strategies to achieve both interpretability and good performance.

In particular, we argue that composition rules for differential privacy, which start from the building blocks and combine them bottom-up, may not offer sufficient flexibility. We suggest an alternative approach, where privacy requirements are specified top-down and implementation choices, such as the allocation of "privacy budget" to several components or the choice between more costly multi-party computing and less accurate noisy data sharing, are optimized afterward.

We start in Section 2 with a brief review of relevant literature and a discussion of the advantages and drawbacks of several strategies. We then sketch our ideas in Section 3 and provide a number of examples to illustrate them.

2 Existing approaches

An important notion of privacy is differential privacy [3]:

Definition 1 (Differential privacy). *Let $\varepsilon > 0, \delta \geq 0$. A (randomized) protocol $\mathcal{A}$ is (ε, δ)-differentially private if for all neighboring datasets X, X', i.e., datasets differing only in a single data point, and for all sets of possible outputs $\mathcal{O}$, we have:*

$$Pr(\mathcal{A}(X) \in \mathcal{O}) \leq e^{\varepsilon} Pr(\mathcal{A}(X') \in \mathcal{O}) + \delta. \tag{1}$$

Several variants and generalizations of differential privacy have been proposed, including proposals focusing on the adversarial model [4] and proposals allowing for more refined secret definitions [10]. In this paper, we will sometimes adopt the term 'secret' from Pufferfish privacy [10] to refer to variables that are private but are not necessary on the level of a single individual in a database of individuals (classic differential privacy).

A wide variety of languages have been proposed to describe privacy properties of systems. Some are aimed at compilers or circuit evaluators [6], others are not necessarily aimed at privacy-preserving technology but rather at trust or consent [8]. In the sequel, we will focus our discussion on languages aimed at specifying privacy properties of systems using privacy-preserving technology.

A classical approach to study the privacy of a compound system is to take the different components as input and to analyze the behavior of the compound system. One basic strategy is to apply composition rules for differential privacy. The basic rule states that if data is queried twice, once with an (ϵ_1, δ_1)-differentially

private algorithm and once with a (ϵ_2, δ_2)-differentially private algorithm, then the combination is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-differentially private. Even though this always holds, this is usually not a tight bound on the privacy of the combination. A number of improved bounds have been proposed, e.g., [5], but even those are often not immediately practical. One issue is that the order of steps to be performed in a system may not be known a priori, e.g., a system could branch due to an if-then-else decision, or parts could be repeated.

To address this problem and at the same time have a more uniform way to represent privacy properties, many authors have proposed languages to specify privacy properties, together with associated techniques to verify whether these properties are satisfied when a number of rules can be applied [11, 9, 7]. The advantage is that next to a language there is a system which can attempt to verify whether a given system satisfies the described privacy properties. However, several problems remain. First, theorem proving style techniques usually only work for a limited set of rules or reasoning primitives and they don't scale very well with increasing problem size. Second, while verifying that a property holds is interesting, optimizing the performance would be even better. In such theorem proving settings, it remains the task of the human expert to design the characteristics of the individual components of the system and combine them such that they collaborate efficiently.

3 Privacy constraint optimization

Similar to earlier work discussed in the previous section, our approach starts from a language to describe privacy constraints. However, rather than aiming at verification, we aim at optimization. We propose to first formulate the problem and its privacy requirements in a systematic way, and then to treat these privacy requirements as the constraints of an optimization problem where the objective function is the utility, or conversely the loss. The loss function can incorporate various types of costs, such as the expected error on the output, the computational cost of the resulting system, or its storage cost.

Below, we first sketch at a high level, how problems can be specified. Next, we provide examples of this idea applied to different types of problems. In this position paper, our goal is not to improve some quantifiable performance measures or to solve more difficult problems than before, but rather to illustrate that the idea of constraint programming with privacy requirements has several potentially interesting applications.

3.1 Problem specification

For the purpose of this paper, our main aim is not to develop a complete language allowing for representing as many as possible privacy properties (languages to describe privacy properties have been proposed in the literature to some extent), our main objective is to open the discussion on how to optimize the performance of a system given fixed global privacy requirements.

Therefore, we will use here only a basic language sufficient for our examples. In particular, we distinguish the following components in a specification:

- **Declaring relevant variables.** We will treat both data and models as random variables: data may be public or private and may be drawn jointly with other data variables from some distribution. According to the definition of differential privacy, a differentially private learning algorithm must be a randomized algorithm that outputs a model which follows some probability distribution conditioned on the training data.
- **Specifying relations between variables (background knowledge).** After specifying the relevant random variables, we can specify the conditional dependencies between these variables using a probabilistic model, e.g., a Bayesian network or a Markov random field.
- **Privacy requirements.** Then, we can specify the required privacy properties. This typically involves specifying that the several possible values of a secret can't be distinguished with significant probability by parties not authorized to know the secret.

Below, we present a number of example scenarios where we can apply the proposed technique of compiling privacy requirements to constraint programs.

3.2 Optimizing differential privacy noise as a function of the desired output

Assume we have n sensitive input variables and we want to answer m queries over these sensitive variables. For the simplicity of our presentation, we will assume that the answer to each query is a linear combination of the sensitive variables. We can specify our problem as follows:

Variables:
- $x \in [0,1]^n$: input
- $y \in \mathbb{R}^m$: intermediate variable
- $A \in \mathbb{R}^{m \times n}$: constant
- $b \in \mathbb{R}^m$: constant
- $\mathcal{O}$: output

Background knowledge:
- $y = Ax + b$
- Loss function: $L = \|\mathcal{O} - y\|_2^2$

Privacy requirements:
- $\mathcal{O}$ is (ϵ, δ)-DP w.r.t. x.

Here, we organize our specification as outlined in Section 3.1. It is clear that revealing the exact answers to the queries y_i is unacceptable, resulting in some approximation of the original answers to the queries. So, we specify a loss function representing the cost of the approximation errors.

There are two classic approaches. First, one can use Local Differential Privacy (LDP) [2]. This means that to every input variable x_i sufficient noise is added to obtain a fully private version $\hat{x}_i$. Next, any query can be answered starting from these noisy versions, so, the output can be computed as $\mathcal{O} = A\hat{x} + b$. A major drawback of this approach is that this requires a lot of noise and hence the expected loss will be high. For the simplicity of our analysis, we use the Gaussian mechanism throughout this example. We get

$$\mathbb{E}\left[L_{LDP}\right] = tr(A^\top A)\left(\frac{2\log(1.25/\delta)}{\epsilon^2}\right).$$

Second, one can use classic differential privacy for each query y_i separately, adding noise to every y_i to obtain a noise version $\hat{y}_i$. If multiple queries are obtained, the realized loss may be still high, and if $m > n$, even higher than in the LDP case above:

$$L_{DP} = \left(\sum_{j=1}^{m}\left(\sum_{i=1}^{n}|A_{i,j}|\right)^2\right)\left(\frac{2\log(1.25/\delta)}{\epsilon^2}\right).$$

In contrast, given the specifications above, we propose to (semi-automatically) generate options to address the privacy requirements, not committing to adding noise to input (as in LDP) or output (as in classic DP), but to address the possibility to add noise at meaningful points in the computation. This could result in the following constraint optimization program:

Minimize

$$\mathbb{E}_{\eta,\xi}\left[L\left(\sigma_{(\eta)},\sigma_{(\xi)}\right)\right] = \mathbb{E}_{\eta,\xi}\left[\|\mathcal{O} - (Ax+b)\|_2^2\right]$$

Subject to

- $\hat{x} = x + \eta$
- $\mathcal{O} = \hat{y} = A\hat{x} + b + \xi$
- $\eta_i \sim \mathcal{N}\left(0, \sigma^2_{(\eta),i}\right)$
- $\xi_i \sim \mathcal{N}\left(0, \sigma^2_{(\xi),i}\right)$
- $\mathcal{O}$ is (ϵ, δ)-DP w.r.t. x.

For one query ($m = 1$), the optimal solution to this problem will correspond with classic differential privacy, in the case the number of queries m is large, the solution of the optimization problem will converge to the local differential privacy case. Between the two extremes, we expect a loss that is lower than either of both classic strategies.

The constraint program we consider is easy to solve numerically, and if the approximations are made which are commonly used for the Gaussian mechanism, we get a relaxed constraint optimization problem only involving quadratic functions.

We hence distinguish four steps to address problems with privacy requirements:

1. Specifying the problem and the privacy requirements
2. Adding options to realize the privacy requirements and casting it into a constraint optimization problem
3. Solving the constraint optimization problem
4. Executing the algorithm with the obtained solutions and parameters

3.3 Shaping differential privacy noise

In a number of situations, the classic additive noise mechanisms don't provide an adequate solution. Consider for example the following problem. Suppose that we have a finite domain $\mathcal{X}$ of positive numbers. Consider n parties numbered from 1 to n. Each party i has a sensitive value $x_i \in \mathcal{X}$ which it doesn't want to be revealed. At the same time, the parties collectively would like to compute $k \geq 2$ means of their private values, including the arithmetic mean m_1 and harmonic mean m_{-1} where

$$m_p = \left(\frac{1}{n} \sum_{i=1}^{n} x_i^p \right)^{1/p}.$$

This gives us the following problem:

Variables:
- $x \in [0,1]^n$: input
- $p \in \mathbb{R}^k$: constant
- $\mathcal{O}$: output

Background knowledge:
- Loss function: $L = \sum_{i=1}^{k} \left(\mathcal{O}_i - m_{p_i}(x)\right)^2$

Privacy requirements:
- $\mathcal{O}$ is (ϵ, δ)-DP w.r.t. x.

The several parties don't trust a common curator and therefore decide they will all share noisy versions $\hat{x}_i$ of their private values x_i and perform the computation on these noisy values. This is also the setting considered by local differential privacy.

Classic additive noise mechanisms such as the Laplace mechanism and the Gaussian mechanism have the drawback that the noise distribution has an infinite domain. So, for every value x_i, especially if x_i is one of the smallest elements of $\mathcal{X}$, there is a probability that $\hat{x}_i$ is close to 0 or negative, which would make the estimation of m_{-1} very difficult.

Several solutions are conceivable. First, for every i we could approximate the p_i-mean $m_{p_i}(x)$ using a separate noisy version of x^{p_i}. Averaging over values of x^{p_i} with additive zero-mean (Laplacian or Gaussian) error would give an unbiased estimate. Still, this would imply that the k means to be computed should share

the available privacy budget. We could follow an approach similar to the one in Section 3.2 to optimally spread the privacy budget.

Another option is to use the full privacy budget for a single noisy version $\hat{x}_i$ of x_i for every i. As we can't use classical additive noise mechanisms, we consider an arbitrary parameterized distribution, and aim at estimating optimal parameters for it subject to a number of desirable properties. We should take into account that if the smallest (respectively largest) possible noisy versions of the x_i can't be much smaller (resp. larger) than the smallest (resp. largest) possible value of $\mathcal{X}$ (in our case to avoid zero or negative noisy versions which may harm the approximation of $m_{-1}(x)$), then we can't use zero-mean additive noise. A common solution is to chose for $\hat{x}_i$ with probability α, an unbiased estimator of x and with probability $1-\alpha$, some background distribution B.

In particular, we consider a distribution over a domain $\mathcal{Y} \supseteq \mathcal{X}$. For $(x, y) \in \mathcal{X} \times \mathcal{Y}$, let $f_{x,y} = P(\hat{x}_i = y \mid x)$, i.e., $f_{x,y}$ is the probability, given that a private value is x that the noise version is y. This naturally leads to the following quadratic program:

Minimize

$$\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} f_{x,y}(x-y)^2$$

Subject to

- $\forall x, \sum_{y \in \mathcal{Y}} f_{x,y} y = \alpha x + (1-\alpha)\mathbb{E}[B]$
- $\forall x, \sum_{y \in \mathcal{Y}} f_{x,y} y^{-1} = \alpha x^{-1} + (1-\alpha)\mathbb{E}\left[B^{-1}\right]$
- $\forall x, \sum_{y \in \mathcal{Y}} f_{x,y} = 1$
- $\mathcal{O}$ is (ϵ, δ)-DP w.r.t. x, i.e., $\forall x_1, x_2, y : f_{x_1,y} \leq e^{\epsilon} f_{x_2,y} + \frac{\delta}{|\mathcal{X}|}$

3.4 Combining building blocks

The examples in Sections 3.2 and 3.3 focused on isolated problems. Practical systems are often large and consist of many steps. Even if for each of these steps a privacy-preserving solution is available, these still need to be combined into a global solution.

Classic approaches to differential privacy often use combination rules, e.g., the combination of an (ϵ_1, δ_1)-differentially private step and an (ϵ_2, δ_2)-differentially private step is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-differentially private. The disadvantage is that the privacy budget is not recycled and therefore is exhausted quickly.

The same approach can be taken using constraint programs. However, additionally, we can attempt to obtain globally better solutions. First, we can combine two constraint programs, which share variables and/or constraints. An optimal solution for the combined program may be globally more optimal than the combination of the solutions of the individual programs. Second, it becomes easier to program design choices. Often, several possible solution strategies exist, especially when considering in distributed settings the trade-off between encryption (which is more expensive in computational cost) or adding noise (which

decreases the utility of the output). In such situations, we can introduce both solution strategies as separate sub-programs of the larger constraint program, and introduce an additional variable π which is 0 when the first solution is used and 1 if the other solution is used. While constraint optimization typically works with real-valued variables, if the constraint programs corresponding to the two solutions don't share parameters then the design choice variable π will be either 0 or 1 in the optimum of the constraint program. In this way, in several cases, the user can focus on specifying requirements and possible solution strategies, while the optimization algorithm computes the value of the alternatives and selects the best solution.

4 Discussion and conclusions

In this position paper, we argue that the explainability of privacy-preserving systems can be helped by clearly specifying the privacy guarantees the systems satisfy, and we propose to see these privacy requirements as constraints in an optimization problem.

First, during the design and development phase, this methodology helps the developer to focus on the requirements rather than on implementation choices. In fact, the constraint optimization problem represents the space of all possible high-level implementations, and the solver accordingly finds the most interesting implementation strategy.

Second, in the deployment phase, such an explicit representation of the privacy guarantees facilitates answering user queries about exactly to what extent sensitive data is protected.

We presented a few examples showing that in several cases the translation of privacy requirements to constraint optimization problems is reasonably easy, and often yields constraint optimization problems which can be solved efficiently. Of course, this doesn't constitute a proof that such a methodology will deliver good results in all cases. An interesting line of future work is to explore more different situations and analyze whether the obtained constraint optimization problems remain tractable and scale well with the problem complexity.

Another idea for future work may be to explore whether this methodology also allows us to translate interpretable fairness requirements to efficiently solvable constraint optimization problems. However, a number of additional challenges that may arise there, e.g., there is no widespread consensus on a single good notion of fairness (as is the case with differential privacy in the privacy domain). Second, while in the current paper on privacy we rely on the relation between uncertainty (e.g., variance) and privacy strength, which often leads to efficiently solvable constraints, it is not immediately clear whether we could rely on a similar relation in the fairness domain.

References

1. European General Data Protection Regulation, https://gdpr-info.eu/

2. Duchi, J.C., Jordan, M.I., Wainwright, M.J.: Local privacy and statistical minimax rates. In: FOCS (2013)
3. Dwork, C.: Differential Privacy: A Survey of Results. In: Agrawal, M., Du, D., Duan, Z., Li, A. (eds.) Theory and Applications of Models of Computation. Springer Berlin Heidelberg (2008)
4. Erlingsson, U., Feldman, V., Mironov, I., Raghunathan, A., Talwar, K.: Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In: SODA (2019)
5. Kairouz, P., Oh, S., Viswanath, P.: The composition theorem for differential privacy. In: Proceedings of the 32nd International Conference on Machine Learning. Lille, France (2015)
6. Kreuter, B., shelat, a.: Lessons learned with pcf: Scaling secure computation. In: Proceedings of the First ACM Workshop on Language Support for Privacy-Enhancing Technologies. p. 7–10. PETShop '13, Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2517872.2517877, https://doi.org/10.1145/2517872.2517877
7. Near, J.P., Darais, D., Abuah, C., Stevens, T., Gaddamadugu, P., Wang, L., Somani, N., Zhang, M., Sharma, N., Shan, A., Song, D.: Duet: An Expressive Higher-order Language and Linear Type System for Statically Enforcing Differential Privacy. Proc. ACM Program. Lang. **3**(OOPSLA), 172:1–172:30 (Oct 2019). https://doi.org/10.1145/3360598, http://doi.acm.org/10.1145/3360598
8. Pardo, R., Le Métayer, D.: Analysis of Privacy Policies to Enhance Informed Consent. In: Foley, S.N. (ed.) Data and Applications Security and Privacy XXXIII. Springer International Publishing (2019)
9. Paverd, A.J., Martin, A., Brown, I.: Modelling and automatically analysing privacy properties for honest-but-curious adversaries. Tech. Rep. (2014), https://www.cs.ox.ac.uk/people/andrew.paverd/casper/casper-privacy-report.pdf
10. Song, S., Wang, Y., Chaudhuri, K.: Pufferfish Privacy Mechanisms for Correlated Data. In: Proceedings of the 2017 ACM International Conference on Management of Data. SIGMOD '17, Association for Computing Machinery (2017)
11. Xu, L.: Modular Reasoning about Differential Privacy in a Probabilistic Process Calculus. In: Palamidessi, C., Ryan, M.D. (eds.) Trustworthy Global Computing. Springer Berlin Heidelberg (2013)